Stack and Subroutines

The Stack

- The stack is a group of memory location in the R/W memory that is used for temporary storage of binary information during the execution of a program.
- The stack is a LIFO (Last in First out) structure.
- The starting location of the stack is defined by loading a 16-bit address into the stack pointer that space is reserved, usually at the top of the memory map.

- The stack normally grows backwards into memory.
- The stack can be initialized anywhere in the user memory map, but stack is initialized at the highest memory location so that there will not be any interface with the program.
- In 8085 microprocessor system the beginning of the stack is defined in the program by using the instruction.
  - ✔ LXI SP, 16 bit.
  - ✔ The LXI SP, a 16 bit state that load the 16 bit address into the stack pointer register.

Information is stored and retrieved from the stack

- The 8085 provide two instruction PUSH and POP for storing information on the stack and retrieving it back.
- Information in the register pairs stored on the stack in reverse order by using the instruction PUSH.
- Information retrieved from the stack by using the instruction POP.
- PUSH and POP both instruction works with register pairs only.
- The storage and retrieval of the content of registers on the stack follows the LIFO (Last in first out) sequence.
- Information in the stack location may not be destroyed until new information is stored in that memory location.

The PUSH instruction

2000 LXI SP, 2099H Load the stack pointer register with the address 2099

2003 LXI H, 42F2H Loads data in the HL register pair

2006 PUSH H The content of the HL register pair pushed into stack

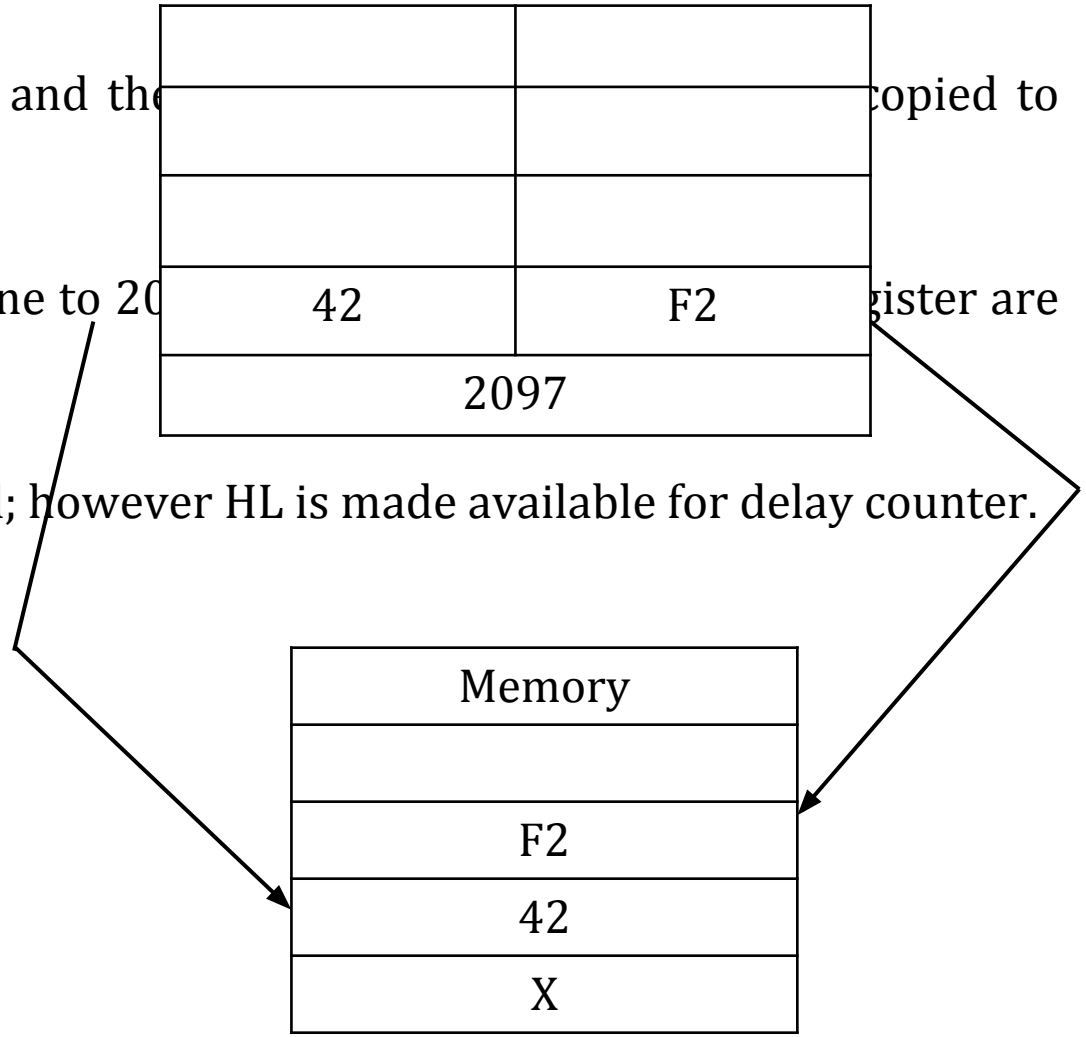
2007 DELAY COUNTER

200F

2010 POP H Saved data in stack pointer register to HL register pair

PUSH H

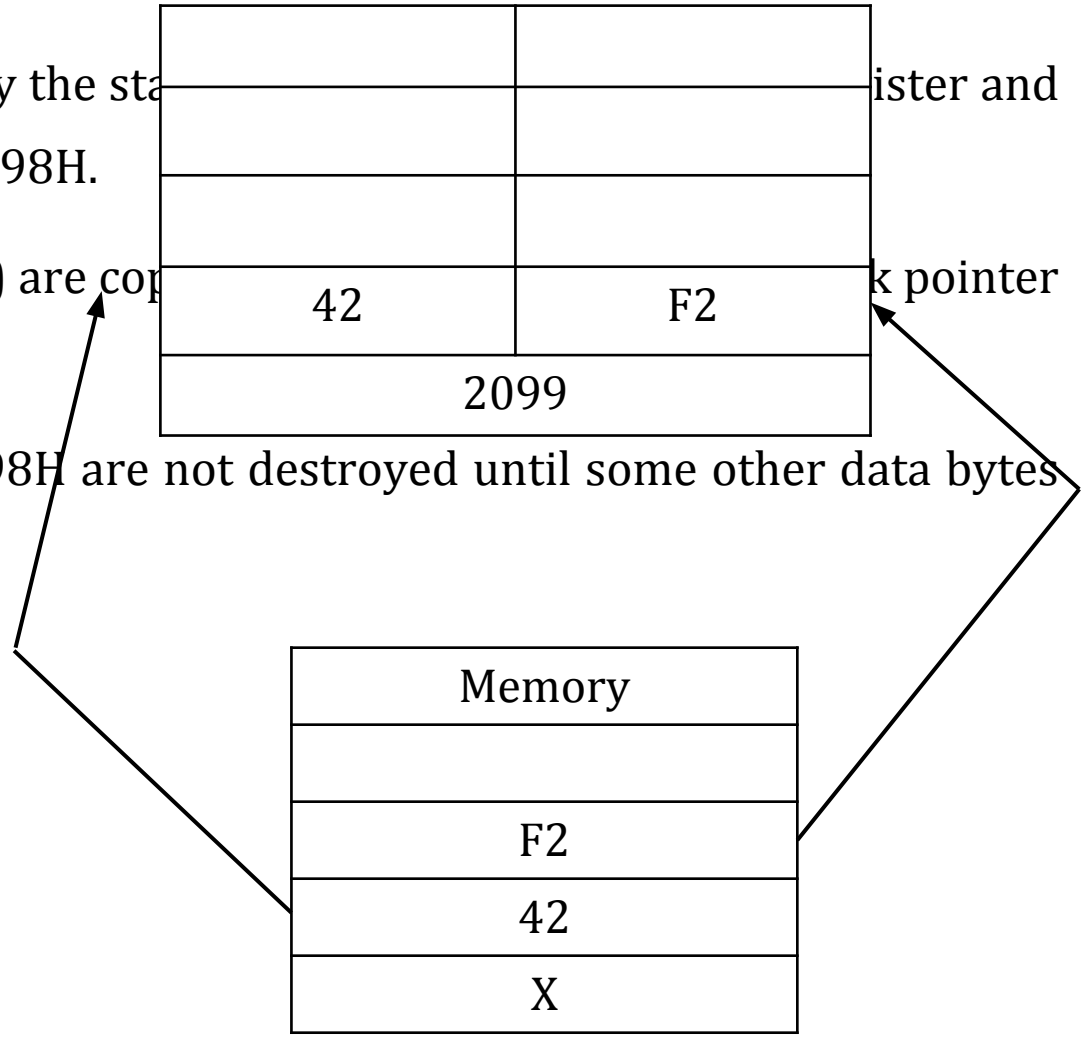
- The stack pointer is decremented by one to 2098H and the [illegible]opied to memory location 2098H.
- The stack pointer register is again decremented by one to 20[illegible]gister are copied to memory location 2097H.
- The contents of the register pair HL are not destroyed; however HL is made available for delay counter.



A	F
B	C
D	E
H	L
SP	

POP H

- The contents of the top of the stack location shown by the sta ister and the stack pointer register is incremented by one to 2098H.
- The contents of the top of the stack (now it is 2098 H) are cop k pointer is incremented by one.
- The contents of the memory location 2097H and 2098H are not destroyed until some other data bytes are stored in these location.



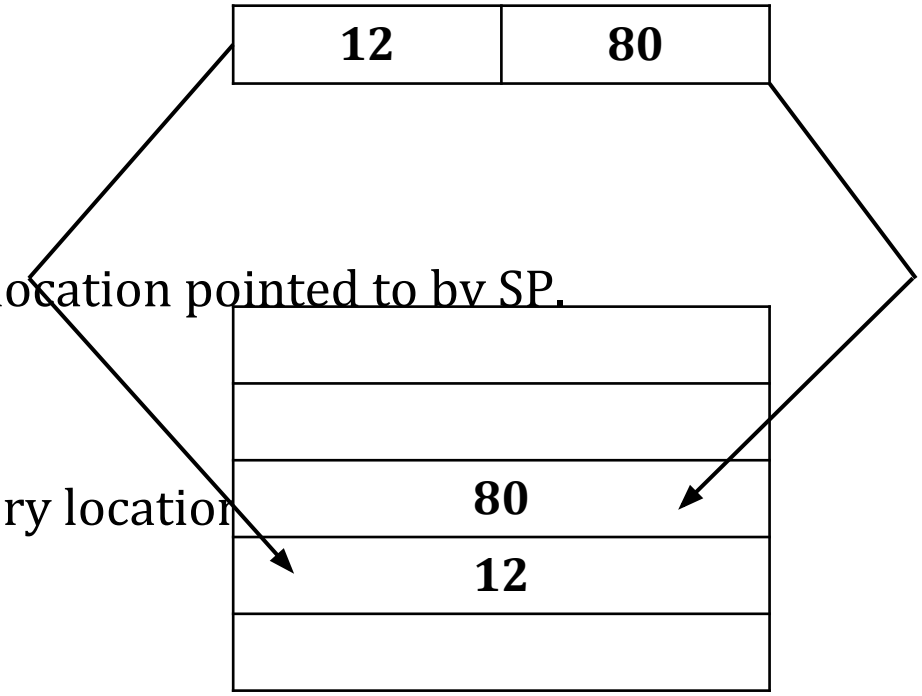
A	F
B	C
D	E
H	L
SP	

Operation of the stack

- During pushing the stack operates in a “decrement then store” style.
  - ✔ The stack pointer is decremented first, then the information is placed on the stack.
- During popping the stack operates in a “use then increment” style.
  - ✔ The information is retrieved from the top of the stack and then the pointer is incremented.
- The SP pointer always points to “the top of the stack”.

PUSH PSW register pair

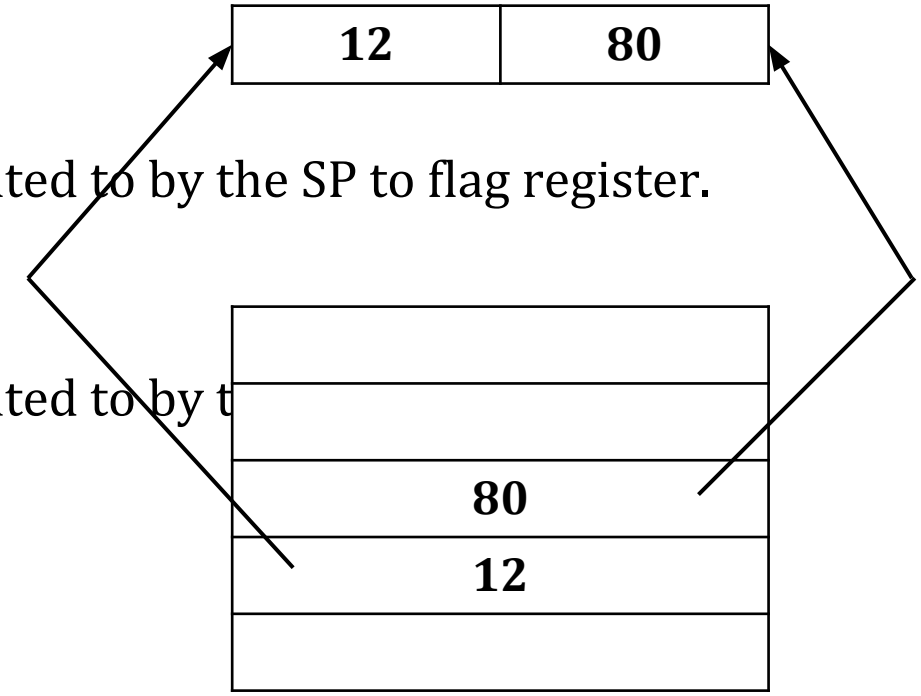
- PUSH PSW (1 byte instruction)
  - ✔ Decrement SP
  - ✔ Copy the contents of register A to the memory location pointed to by SP.
  - ✔ Decrement SP.
  - ✔ Copy the contents of Flag register to the memory location



A Flag

POP PSW register pair

- POP PSW (1 byte instruction)
  - ✔ Copy the contents of the memory location pointed to by the SP to flag register.
  - ✔ Increment SP.
  - ✔ Copy the contents of the memory location pointed to by t
  - ✔ Increment SP.



A Flag

Subroutine in 8085

- In computers, a subroutine is a sequence of program instructions that perform a specific task, packaged as a unit. This unit can then be used in programs wherever that particular task have to be performed. A subroutine is often coded so that it can be started (called) several times and from several places during one execution of the program, including from other subroutines, and then branch back (return) to the next instruction after the call, once the subroutine's task is done. It is implemented by using Call and Return instructions. The different types of subroutine instructions are

Unconditional Call instruction –

- CALL address is the format for unconditional call instruction. After execution of this instruction program control is transferred to a sub-routine whose starting address is specified in the instruction. Value of PC (Program Counter) is transferred to the memory stack and value of SP (Stack Pointer) is decremented by 2.

Conditional Call instruction –

In these instructions program control is transferred to subroutine and value of PC is pushed into stack only if condition is satisfied.

INSTRUCTION	PARAMETER	COMMENT
CC	16-bit address	Call at address if cy (carry flag) = 1
CNC	16-bit address	Call at address if cy (carry flag) = 0
CZ	16-bit address	Call at address if ZF (zero flag) = 1
CNZ	16-bit address	Call at address if ZF (zero flag) = 0
CPE	16-bit address	Call at address if PF (parity flag) = 1
CPO	16-bit address	Call at address if PF (parity flag) = 0
CN	16-bit address	Call at address if SF (signed flag) = 1
CP	16-bit address	Call at address if SF (signed flag) = 0

Unconditional Return instruction –

- RET is the instruction used to mark the end of sub-routine. It has no parameter. After execution of this instruction program control is transferred back to main program from where it had stopped. Value of PC (Program Counter) is retrieved from the memory stack and value of SP (Stack Pointer) is incremented by 2.

Conditional Return instruction –

- By these instructions program control is transferred back to main program and value of PC is popped from stack only if condition is satisfied. There is no parameter for return instruction.

INSTRUCTION	COMMENT
RC	Return from subroutine if cy (carry flag) = 1
RNC	Return from subroutine if cy (carry flag) = 0
RZ	Return from subroutine if ZF (zero flag) = 1
RNZ	Return from subroutine if ZF (zero flag) = 0
RPE	Return from subroutine if PF (parity flag) = 1
RPO	Return from subroutine if PF (parity flag) = 0
RN	Return from subroutine if SF (signed flag) = 1
RP	Return from subroutine if SF (signed flag) = 0

Advantages of Subroutine –

1.Decomposing a complex programming task into simpler steps.

2.Reducing duplicate code within a program.

3.Enabling reuse of code across multiple programs.

4.Improving tractability or makes debugging of a program easy.

Subroutines

- A subroutine is a group of instruction written separately from the main program to perform a function that occurs repeatedly in the main program.
  - ✔ when a main program calls a subroutine the program execution is transferred to the subroutine after the completion of the subroutine, the program execution returns to the main program.
  - ✔ The microprocessor uses the stack to store the return address of the subroutine.

Subroutines

- The 8085 has two instructions for dealing with subroutines.
  - ✔ The CALL instruction is used to redirect program execution to the subroutine.
  - ✔ The RET instruction is used to return to the main program at the end of the subroutine.

The CALL instruction

- CALL, 16 bit
  - ✔ Call subroutine is conditionally located at the memory address specified by the 16 bit operand.
  - ✔ This instruction places the address of the next instruction on the stack and transfer the program execution to the subroutine address.

The RET instruction

- Return unconditionally from the subroutine.
- This instruction locates the return address on the top of the stack and transfers the program execution back to the calling program.

General characteristics of CALL and RTE instruction

- The CALL instructions are 3-byte instruction; the second byte specified the low order byte and the third byte specifies the high order byte of the subroutine address.
- The Return instructions are 1-byte instruction.
- A CALL instruction must be used in conjunction with a Return instruction in the subroutine.

Necessary steps to implement a subroutine

- The stack pointer register must be initialized, preferably at the highest memory location of the R/W memory.
- The CALL instruction should be used in the main program accompanied by the RET instruction in the subroutine.

Conditional call and RTE instructions

- The 8085 supports conditional call and conditional RTE instructions.
  - ✔ The same conditions used with conditional JUMP instructions can be used.
  - ✔ CC, call subroutine if carry flag is set.
  - ✔ CNC, call subroutine if carry flag is not set.
  - ✔ RC, return from subroutine if carry flag is set.
  - ✔ RNC, return from subroutine if carry flag is not set.

Ref.: https://www.slideshare.net/safinbiswas/stack-in-microprocessor-8085presentation

https://www.geeksforgeeks.org/subroutine-in-8085/